An area-efficient 2-D convolution implementation on FPGA for space applications

An area-efficient 2-D convolution implementation on FPGA for space applications

Authors: Di Carlo S., Gambardella G., Indaco M., Rolfo D., Tiotto G., Prinetto P.,

Published in the Proceedings of the IEEE 6th International Design and Test Workshop (IDT), 11-14 Dec. 2011, Beirut, LI.

DOI: 10.1109/IDT.2011.6123108

An Area-Efficient 2-D Convolution Implementation on FPGA for Space Applications

Stefano Di Carlo*, Giulio Gambardella†, Marco Indaco*, Daniele Rolfo†, Gabriele Tiotto*, Paolo Prinetto*
†CINI
Via Ariosto 25, 00185 Roma, Italy
Email: {*FirstName.LastName*}@consorzio-cini.it
*Politecnico di Torino
Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi 24, I-10129, Torino, Italy
Email: {*FirstName.LastName*}@polito.it

***Abstract*—The 2-D Convolution is an algorithm widely used in image and video processing. Although its computation is simple, its implementation requires a high computational power and an intensive use of memory. Field Programmable Gate Arrays (FPGA) architectures were proposed to accelerate calculations of 2-D Convolution and the use of buffers implemented on FPGAs are used to avoid direct memory access. In this paper we present an implementation of the 2-D Convolution algorithm on a FPGA architecture designed to support this operation in space applications. This proposed solution dramatically decreases the area needed keeping good performance, making it appropriate for embedded systems in critical space applications.**

I. Introduction

Image analysis is a fundamental task in remote sensing applications and makes it possible to enhance raw images acquired from camera sensors. Several techniques have been developed in image processing for enhancing images obtained from space probes during missions with critical requirements. In space applications, remote sensing satellites and spacecrafts modules have limited onboard computing capacity due to the external environment conditions.

Several image processing systems for space applications take effort of one or more filtering algorithms implemented sequentially in a filtering chain. One of the main steps of the filtering chain is a 2-D discrete convolution algorithm [1] [2]. This is conceptually a simple process performing a sum of products of constant values (kernel matrix) by variable values (image matrix). However, its software implementation requires a huge amount of resources in terms of computational power, latency and power consumption. This can result in significant drawbacks when targeting embedded systems for real-time space applications. Therefore, its hardware implementation (e.g., with reconfigurable devices as FPGA) must aim at increasing performances exploiting the natural parallelism of 2D convolution algorithms. Although there are many examples in literature of 2D convolution implementations, few of them take into account the area occupation constraint as the main requirement.

Proposed solutions for 2D convolution on FPGA mainly focus on different buffering models for performance optimization in terms of bandwidth and data transfer to/from the memory [3] [4] [5] [6] [7]. They also focus on strategies to increase the throughput in terms of frequency. The adopted buffering technique has an impact on the number of memory accesses required to read the same pixel. This relies on the need of using the same pixel to perform several convolution operations on contiguous pixels. The Full Buffering technique has been implemented in [8]. When computing an image of MxN pixel, this approach requires the buffering of a whole row of M-1 pixels. The data buffers are used as delay lines to shift the rows until the number of loaded pixels is enough to perform the convolution. This strategy is inefficient when the area is a constraint, since it requires a high number of buffers that dramatically increase when a high size image processing is required [8] [9] [4].

More efficient solutions are the Partial Buffering (PB) [9], which decreases the number of required buffers by increasing the bandwidth required for the memory storage. In this solution each pixel is read M times.

The hybrid schemes called MultiWindow Partial Buffering (MWPB) [3] aims at implementing area-efficient solutions and at keeping the memory bandwidth lower with respect to previous solutions. MWPB is based on the implementation of a shift register matrix of size greater than the kernel matrix. In this way it is possible to share the same pixel with the different convolution windows and thus keeping the memory access frequency lower. In [4], three solutions are described, that take effort of a PB approach. This PB approach has been modified in order to be area efficient but with a high memory bandwidth. In [10] an implementation of a module for 2-D Convolution based on a logarithmic approach is described. This approach aims at minimizing the power consumption. The main drawback of logarithmic approach is the introduction of approximation errors. The architecture presented in [5] exploits symmetric kernels and performs the computation in the logarithmic domain, to avoid the use of multipliers. [7] presents a systolic configurable architecture for image filtering. The core of the architecture is a 2D systolic array based on particular processing elements called Configurable Window Processor. These elements can be configured on the basis of the type of the required filtering.

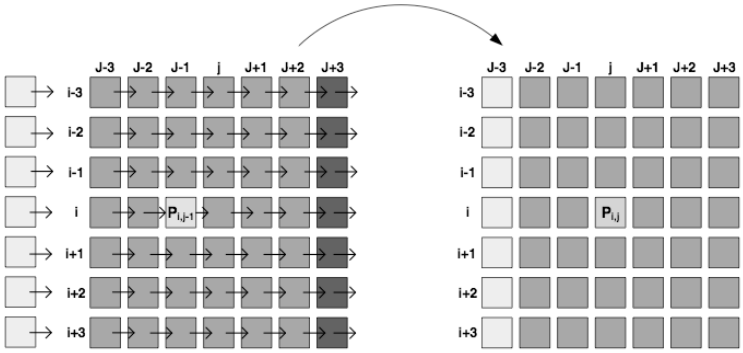

Fig. 1. Right to left moving window

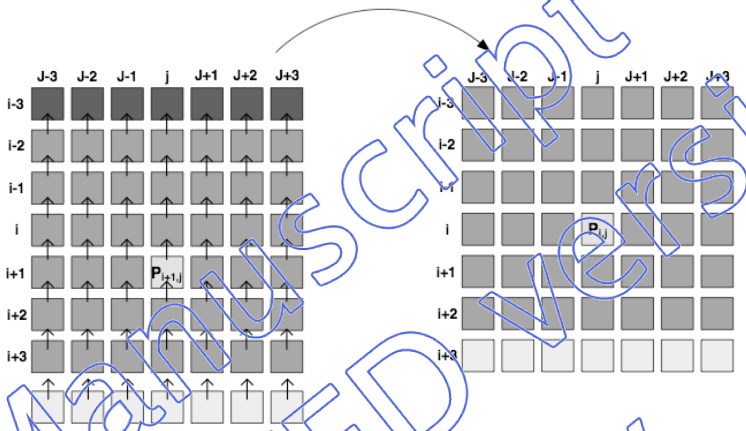

Fig. 2. Top to down moving window

In this paper we propose the implementation of a bidimensional convolver on FPGA. Our implementation is based on a novel approach that is, at the same time, area-efficient and keeps low memory bandwidth. It is one key stage of the filter chain that will support the Entry Descending and Landing (EDL) vision based approach.

The paper is organized as follows: Section 2 describes the specific application and the proposed approach. Section 3 details our approach. Section 4 describes the architecture of our system. Section 5 shows experimental results. Section 6 concludes the paper and depicts the future work.

II. Real Time FPGA for Space Applications

In this section we detail the constraints and goals of the project. The Entry Descending and Landing (EDL) vision based approach is a technique that allows to automatically drive a space module for landing on planet surface. This operation is performed without any remote user interaction. The major constraints imposed by our EDL target application are:

- Input image constraints:
  - Size of the image: 1024x1024 bits
  - Fixed number of bits per pixel: 8 bits
  - Boundary Condition: Zero padding.
- Kernel Matrix constraints:
  - Size: 7 x 7 pixel with 8 bit for pixel, weighted for the sharpening filter
  - Data representation: 2's complement notation
  - Input matrix stored in external memory.
- System constraints:
  - Memory size: 32 bit
  - Bus length: 32 bit.

Therefore we need to minimize area utilization since we plan to implement the whole filter chain on the FPGA.

The image processing algorithms have been deeply analyzed to identify the task to be allocated to the processor and the co-processor. Considering that the processor is in charge of several different tasks over the image processing (sensor acquisition, actuator control, data handling), the most demanding image processing computational functions will be assigned to the co-processor respectively.

The analysis of different algorithms shows that the most common and demanding operation of the image processing chain is the image filtering. Filters like Sobel [11], Prewitt [12], Gaussian [13], median [14] and other simple filtering operations make a strong use of the 2D convolution operation

on a square window. The different coefficients of the window can also be arranged for the different filter implementations.

This makes the 2D convolution the best candidate as a main co-processor function. Further analysis of the different image filtering functions showed that the best compromise between required processing performances and quality filtered image can be obtained by means of 2D convolution on a maximal kernel window size of 7x7.

III. The Proposed Approach

The 2-D convolution on an input image I is expressed by the well known expression:

$$I' = \sum_i \sum_j w_{i,j} \times I_{m+i,n+j} \forall (i,j) \in R \times S$$

Where I' is the output image and $w_{i,j}$ is the convolution kernel weight. A window RxS centred on a pixel (m, n) is extracted from the input image and each pixel is multiplied by the corresponding kernel weight. The products are then added to produce the output pixel value [3]. Shift registers and FIFO are used to implement the moving window. The use of these devices has a strong impact on the area utilization.

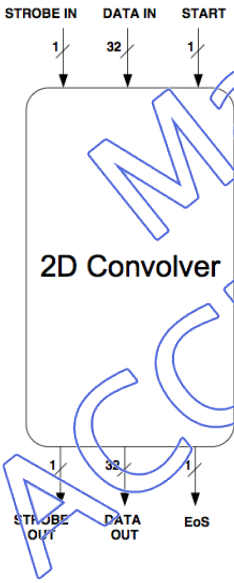

Fig. 3. System's Top Level View Architecture

In our approach we use a KxK moving window with the same size of the kernel matrix. To minimize the area utilization, it is necessary to implement an array of shift registers of width equal to the KxK window size.

When we compute the pixel $P_{i,j}$ where i and j are the row and the column of the image, respectively, the shift registers contain the pixels from i-3 to i+3 and from j-3 to j+3. The pixel computation is performed by rows, from left to right for the even index row and from right to left for odd index rows. Summarizing we move the window in a winding line fashion. Let $window(|i| \leq \lfloor(\frac{K}{2})\rfloor, |j| \leq \lfloor(\frac{K}{2})\rfloor)$ be the needed portion of data for computing the convolution of a single pixel. K is the size of symmetric kernel matrix and new_col and new_row are column vector and a row vector respectively, containing K new pixels.

In order to compute the $P_{i,j+1}$ pixel, the one to the right with respect to the $P_{i,j}$ previously computed, the procedure is the following:

1: **procedure** LoadColumn(j, K, new_col)
2: **for** $x \leftarrow (j + \lfloor(\frac{K}{2})\rfloor), (j - \lfloor(\frac{K}{2})\rfloor) + 1$ **do**
3: $window(*, x) \leftarrow window(*, x - 1)$ ▷ $*$: all rows
4: **end for**
5: $window(*, x - 1) \leftarrow new\_col$
6: **end procedure**
7: **procedure** ComputePixel($window$)
8: ...
9: **end procedure**

The above steps are shown in Figure 1.

In the case of the $P_{i+1,j}$ computation we proceed as follows:

1: **procedure** LoadRow(i, K, new_row)
2: **for** $x \leftarrow (i - \lfloor(\frac{K}{2})\rfloor), (i + \lfloor(\frac{K}{2})\rfloor) - 1$ **do**
3: $window(x, *) \leftarrow window(x + 1, *)$ ▷ $*$: all columns
4: **end for**
5: $window(x + 1, *) \leftarrow new\_row$
6: **end procedure**
7: **procedure** ComputePixel($window$)
8: ...
9: **end procedure**

The above steps are shown in Figure 2.

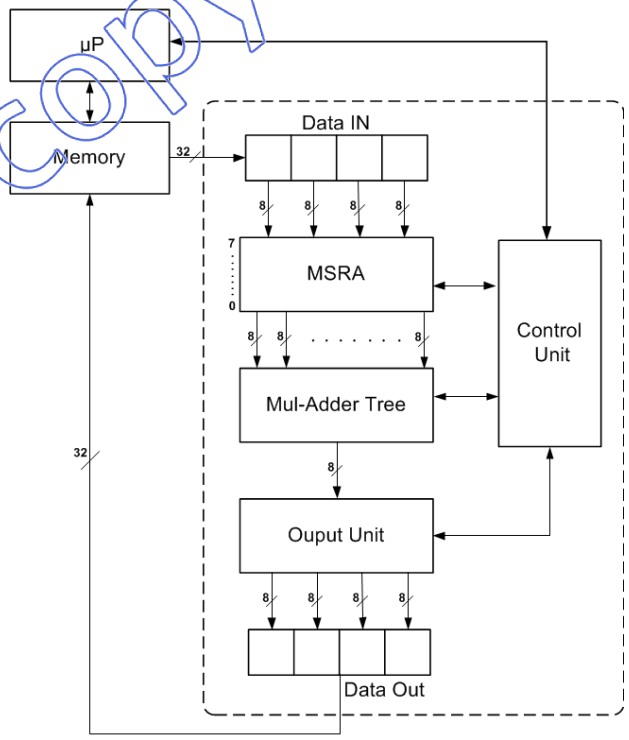

Fig. 4. Detailed system architecture

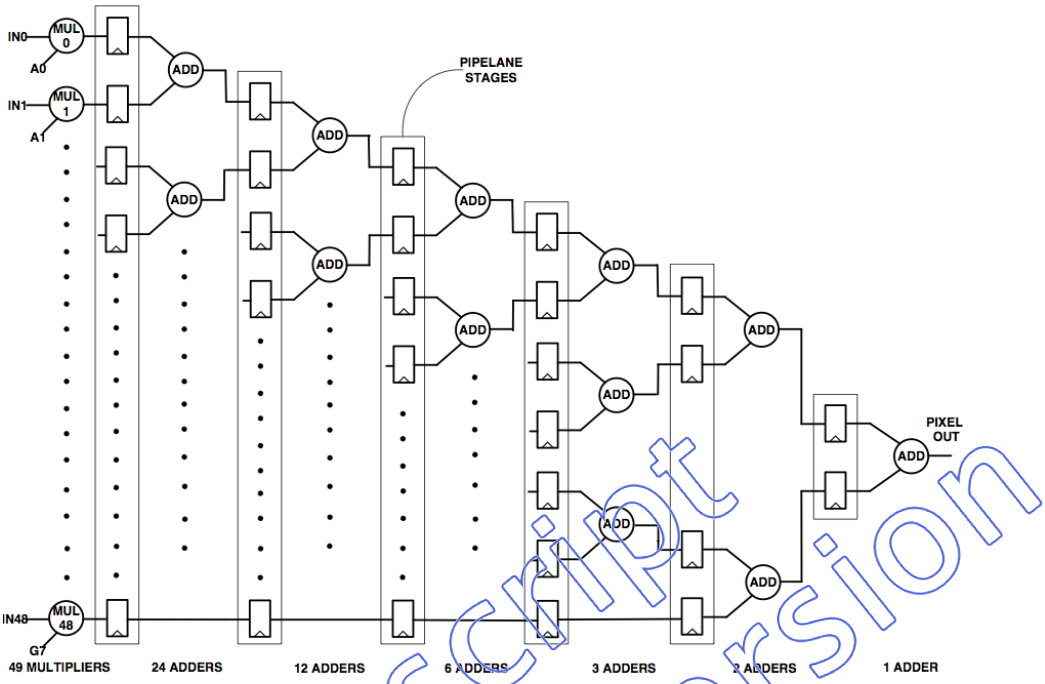

Fig. 5. The Mul-Adder Tree

IV. The System Architecture

The architecture has been designed jointly with the Thales Alenia Space and it's a candidate to be a possible solution. Based on the specific implemented filtering chain, we designed a moving window architecture for 2D convolution shift-based on FPGA under well-defined requirements. Figure 3 shows the system's top level view. When the start signal is asserted, the DMA module loads 4 pixel into the convolver. Strobe In validates input data. When the unit is ready to provide 4 convolved pixel, it asserts Strobe Out signal. Asserting EoS signals to the microprocessor the end of the convolution process.

As underlined in section 2, we focused on optimizing the area occupation. This has an impact on the performance. Nevertheless the number of images processed by the system per second is kept sufficiently high.
The internal structure is composed of 4 blocks (Figure 4):

- Multi-Shift Register Array (MSRA), for implementing moving window,
- Mul-Adder Tree, that performs multiplications and sums,
- Output Unit, that groups the computed pixels and triggers external logic for data storing.
- Control Unit, that manages modules' synchronization.

The MSRA is implemented using a 49 registers array, each register is 8 bit wide, to be able to store a pixel. The MSRA is designed to perform the left/right shifting as described in Section III to simulate the moving window scan along the horizontal direction. Moreover it allows for the up/down shifting along the vertical direction. The output signals of each registry are connected to the Mul-Addder Tree to perform the products with the weights of the kernel matrix.

TABLE I
CPU Architectures and expected performances

Performance (MIPS)	Processor	Co-Processor
[200;300]	LEON3	DSP (on FPGA)
[300;400]	LEON3	LEON3 + DSP (on FPGA)
>400	LEON3	PPC

To optimize the clock frequency we introduced 6 pipeline stages in it. It is worth to point out that the introduction of the pipelined datapath increases the area utilization, nevertheless it is affordable if we take into account the significant improvement in terms of efficiency.

The output unit includes an output buffer to store 4 pixels, and a data normalization unit. The normalization unit is required to normalize the size of the convolved pixel to 8 bit.
The Control Unit coordinates the data stream within the system architecture. Mainly, it drives the control signals needed to shift the MSRA and to acquire a new row/column. Another important functionality is the management of the boundary effects, as specified in Section 3.

We implemented the convolution of the pixels within the frame of the image (first 3 rows and last 3 rows, first 3 columns and last 3 columns) with the zero padding technique.

V. Experimental Results

The FPGA target board is a Virtex 4 xc4vlx25 10sf363 [15] [16] [17].

TABLE II
AMOUNT OF LOGIC USED

Logic Utilization	Used	Available	% of Utilization
Slices	1356	10752	12%
Slice Flip Flop	1804	24192	7%
4 input LUTs	2568	24192	10%

TABLE III
HARDWARE RESOURCES UTILIZATION

Architecture	Slices	Kernel Matrix Size	Area Utilization
Proposed	1356	7x7	12%
MWPB [3]	2290	5x5	21%

Table II shows the area occupation on the target FPGA in terms of slices, flip-flops and LUT's. Table III shows a comparison between our approach and the MWPB proposed in [3]. To the best of our knowledge MWPB is the only approach comparable with our approach from the buffer unit point of view. It worth to point out that our approach provides better results in terms of area even if our kernel matrix size is 7x7 against the 5x5 size in [3] . Therefore our architecture significantly improves area occupation.

VI. CONCLUSION

This paper presented the implementation of a 2-D Convolution algorithm on FPGA for space application. The results show an efficient area overhead compared to previously proposed solutions. Future work will aim at further improving the design and the trade-off between latency and area occupation.

VII. ACKNOWLEDGMENTS

The authors would like to thank the colleagues of the Command Control and Data Handling group of Thales Alenia Space Italia S.p.A. in Turin for the very fruitful collaboration.

REFERENCES

[1] S. Larsen and A.-B. Salberg, "Automatic vehicle counts from quickbird images," in *Urban Remote Sensing Event (JURSE), 2011 Joint*, april 2011, pp. 9 –12.

[2] F. Khalvati, M. Aagaard, and H. Tizhoosh, "Accelerating image processing algorithms based on the reuse of spatial patterns," in *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*, april 2007, pp. 172 –175.

[3] H. Zhang, M. Xia, and G. Hu, "A multiwindow partial buffering scheme for fpga-based 2-d convolvers," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 2, pp. 200–204, 2007.

[4] F. Cardells-Tormo and P.-L. Molinet, "Area-efficient 2-d shift-variant convolvers for fpga-based digital image processing," in *Signal Processing Systems Design and Implementation, 2005. IEEE Workshop on*, nov. 2005, pp. 209 – 213.

[5] M. Zhang and V. Asari, "A fully pipelined multiplierless architecture for 2d convolution with quadrant symmetric kernels," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, dec. 2006, pp. 1559 –1562.

[6] S. Perri, M. Lanuzza, P. Corsonello, and G. Cocorullo, "A high-performance fully reconfigurable fpga-based 2d convolution processor," *Microprocessors and Microsystems*, vol. 29, no. 8-9, pp. 381 – 391, 2005, ¡ce:title¿Special Issue on FPGAs: Case Studies in Computer Vision and Image Processing¡/ce:title¿.

[7] C. Torres-Huitzil and M. Arias-Estrada, "Real-time image processing with a compact fpga-based systolic architecture," *Real-time imaging*, vol. 10, no. 3, pp. 177–187, 2004.

[8] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-d convolvers for fast digital signal processing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 7, no. 3, pp. 299 –308, sept. 1999.

[9] X. Liang, J. Jean, and K. Tomko, "Data buffering and allocation in mapping generalized template matching on reconfigurable systems," *The Journal of Supercomputing*, vol. 19, no. 1, pp. 77–91, 2001.

[10] H. Ngo and V. Asari, "Design of a logarithmic domain 2-d convolver for low power video processing applications," in *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, april 2009, pp. 1280 –1285.

[11] C. Ma, L. Yang, W. Gao, and Z. Liu, "An improved sobel algorithm based on median filter," in *Mechanical and Electronics Engineering (ICMEE), 2010 2nd International Conference on*, vol. 1, aug. 2010, pp. V1–88 –V1–92.

[12] W. Gao, L. Yang, X. Zhang, B. Zhou, and C. Ma, "Based on soft-threshold wavelet de-noising combining with prewitt operator edge detection algorithm," in *Education Technology and Computer (ICETC), 2010 2nd International Conference on*, vol. 5, june 2010, pp. V5–155 –V5–162.

[13] P.-Y. Hsiao, S.-S. Chou, and F.-C. Huang, "Generic 2-d gaussian smoothing filter for noisy image processing," in *TENCON 2007 - 2007 IEEE Region 10 Conference*, 30 2007-nov. 2 2007, pp. 1 –4.

[14] C.-C. Chang, J.-Y. Hsiao, and C.-P. Hsieh, "An adaptive median filter for image denoising," in *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, vol. 2, dec. 2008, pp. 346 –350.

[15] *Virtex-4 FPGA Configuration User Guide*, v1.11 ed., Xilinx Inc., 2009.

[16] *Early Access Partial Reconfiguration User Guide*, v1.2 ed., Xilinx Inc., 2008.

[17] *Virtex 4 Family Overview*, v3.0 ed., Xilinx Inc., 2007.